Report Number: WUCS-95-34

1995-01-01

A Single-Stroke Orientation-Orient Gesture System

Authors: Yike Hu

A Single-Stroke Orientation-Orient Gesture System

Yike Hu

WUCS-95-34

November 1995

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis, MO 63130-4899

A Single-Stroke Orientation-Orient Gesture System

Hu, Yike

1. INTRODUCTION

Accompanying development of pen-based computers, a new tool of human-computer interaction -- the gesture system has been created and put in really use in some products. Compared to the traditional keyboard-based command and mouse-based menu, the pen-based gesture has its unique advantages. The computer users no longer need to memorize a set of complicated command, either to manipulate the uneasy mouse. All they need to do is just to draw some pre-defined simple signs -- the gestures by a pen, the tool they used to use in their daily work and life, the computer will then understand what they mean and perform appropriate action they invoke. Hence, the computers become more humanized.

However, as everything has its advantages and disadvantages, the gesture system could not be absolutely perfect also. The major problem is the recognition rate or accuracy rate. Because of several random factors, the recognition rate could not be 100%. Usually, it is believed that recognition rate in average greater than 90% could be acceptable. But for a particular user, it will be unpleasant while a special gesture has been misrecognized frequently even though in a short period. Being worried about when the misrecognition will happen again, he/she becomes unconfident with the computer and even himself /herself. So it becomes a big job to improve the recognition rate as high as possible.

Usually, people concentrate their attention on improving recognition algorithm to seek higher recognition accuracy. Various approaches, such as the algorithm for character recognition, even neural networks, have been imported into gesture recognition. The prices payed for higher recognition rate are more complicated program, more memory size requirements, higher costs and less respond speed. However, the recognition rate is

depending on not only recognition algorithm, but also the gesture system design itself. A finely-designed gesture system could minimize the complexity of the recognition algorithm and lead to higher recognition rate without increasing resource requirements and system cost. Based on above analysing, author has made some exploring on a single-stroke gesture system and found that it is possible to reach higher recognition rate by using simple recognition algorithm, provided the system design is suitable.

2. SYSTEM DESIGN

2.1 The Design Principles

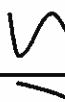
Some common sense guidelines for designing gesture are described as: simplicity, drawability, complexity, rememberability, and intuitiveness. Actually, some of above guideline factors are related to each other. For example, a simple design usually is easy to draw; most easy to draw gestures are also easy to remember. So, at the end, I believe that the key factors of the guidelines remain only two: easy to use (including easy to draw and easy to remember) for human and easy to be recognize by computer. They are independent to each other, yet contradictory in a sense. The task of the designer is to meet both requirements as much as possible.

2.2 The Details of the System Design

2.2.1 The System's Outline

Considering ease of use, the single stroke is the best choice of course. This contributes to easy-recognition also because there will no time out problem. For easier recognition, All the gestures are organized obeying certain rule which is easy to remember also. Hence, the requirements of both the usability and recognizability are met simultaneously.

The whole gesture system is shown as Tab 1. For convenience of calling each gesture, some appropriate ASCII characters similar to the gestures has been assigned.

Turning Point Number / Orientation	0		1		2		3	
	Gesture	ASCII	Gesture	ASCII	Gesture	ASCII	Gesture	ASCII
up-right (north-east)		/		A		N		M
down-right (south-east)	(Tap)	.		V		n		W
down-right (south-east)		\		>		Z		}
down-left (south-west)		(Enter)		<		S		{

* Starting point
** Ending point

Tab 1. Gestures Summary

There are some terms regarding to the design of such a gesture system:

Orientation: refers to the direction along which the stroke of the gesture goes. It could be represented by some numbers as wold be described in **2.2.3** and **2.2.4**

Starting Orientation: refers to the orientation of the gesture stroke between its first (starting) point and first turning point.

Turning Point: refers to that point where the stroke changes its orientation.

2.2.2 The Rule of the Gesture Generating

There is no difficulty of finding out what the rule is by checking the *orientation* and the number of the *turning points* for each stroke. Let's have a look at the Tab 1. There are 16 gestures which have been constructed as a 4x4 matrix. The gestures within the same column have the same number of turning points (from 0 to 3), while the ones within the same row have the same starting orientation. The only exception is the "Tap" which has no orientation and occupies the first position of the second row. So the rule has two

factors: starting orientation of the stroke and the number of the turning points.

2.2.3 The Recognition Code

The rule mentioned above is very helpful to the recognition. First of all, a special code for recognition, let's call it recognition code, is designed incorporating the two factors of the rule. It consists of two fields: Feature code and Turning Point Count, corresponding to the starting orientation of the stroke and the number of the turning points respectively. Where the Feature Code is the value of the numeralized orientation. The format of Recognition Code is shown as Fig 1.

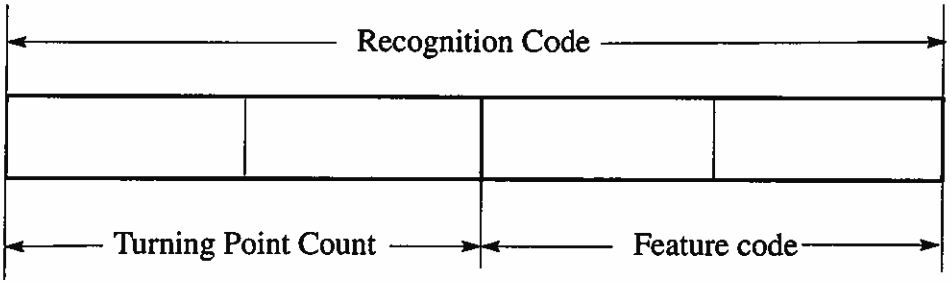


Fig 1 The Format of Recognition Code

2.2.4 The Feature Code

To determine the value of the Feature Code, simply check the polarity of the coordinates' increments Δx and Δy. Fig 2 shows the relationship between the polarity of $(\Delta x, \Delta y)$ and the value of the Feature Code.

Note the orientation of North-West is left unused because it is counteractive to the writing habit of most people. The code for that orientation (01) is assigned to the group of the second row in Tab 1, so that it could be distinguished from the third group of gestures which have the same starting orientation.

Stroke	Orientation	Polarity		Code
		Δy	Δx	
* **	North-East	+	+	00
* **	South-East	−	+	10
** *	South-West	−	−	11

* Starting point
** Ending point

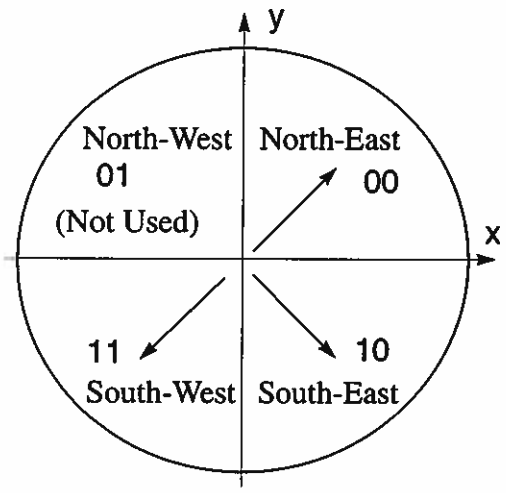


Fig 2. The Definition of the Feature Code

3. RECOGNITION ALGORITHM

3.1 The Basic Recognition

Obviously, the recognition algorithm based on that gesture system could be simplified dramatically. The recognition will be performed synchronously with the gesture drawing. To have a gesture recognized, the the recognizer should:

1) Calculate the increments of the coordinates Δx and Δy.

2) Set the feature code according to the polarities of the increments Δx and Δy. Note the code for the group of the second row in Tab 1 should be set to 01, so that it could be distinguished from the third group of gestures which have the same starting orientation.

3) Detect the turning point and calculate the number of turning points. If either the polarity(sign) of Δx or Δy is found changed, then the turning point count plus 1.

4) Compose the recognition code from the feature code and turning point count:

RC = FC + 4*TC

The flowchart of the recognition procedure is given in Fig 3.

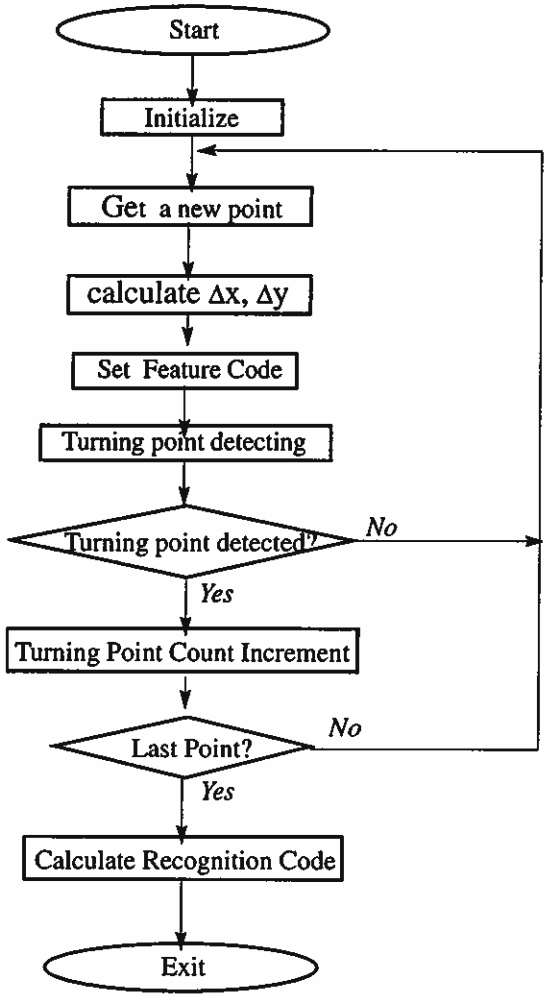
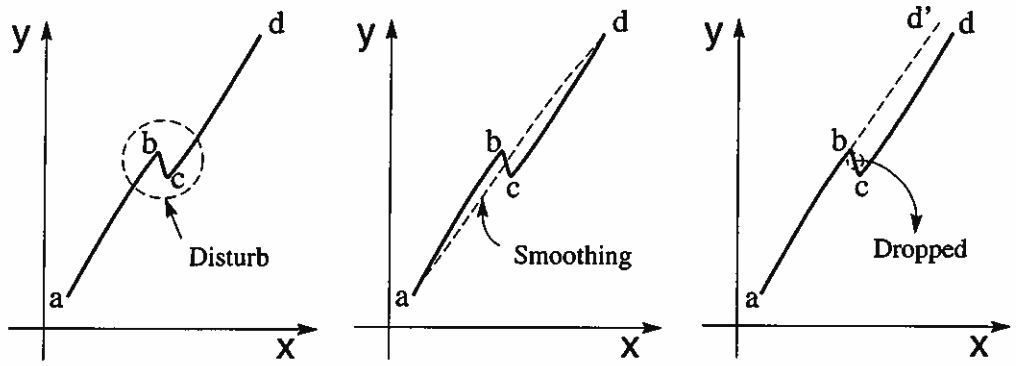


Fig 3. Basic Recognition Flowchart

3.2 The Problem of Disturb and Its Solution

The algorithm discussed above is just a theoretical one. Actually, because of the exist of disturb the gestures could not be drawn perfectly. As shown in Fig 4 (a), suppose the user wanted to draw a "/", but actually there is a disturb happened from point *b* to point *c*. It will be recognized as "N" rather than "/" if do not take any measures to eliminate the disturb. The traditional way to resolve such problem is to "smoothing" the cure line by

replacing the original values of coordinates with the average ones, as shown in Fig 4 (b).



(a) (b) (c)

Fig 4. Disturb and the Methods of Eliminating It

There is another way easier to eliminate the influence of the disturb. Because the length of the disturb (from *b* to *c*) is very short, the fact could be used for detection of disturb. We can calculate the length of a stroke between two turning points, then compare it with a pre-set valve value, if the stroke length between two turning points is found less than that valve value, then the short parts of the stroke is regarded as a disturb and to be dropped. As shown in Fig 4(c), the length of *bc* is found shorter than the valve value, so *bc* has been dropped, and the stroke *a-b-c-d* is replaced with *a-b-d'*.The disturb elimination algorithm in principle is given as Fig 5.

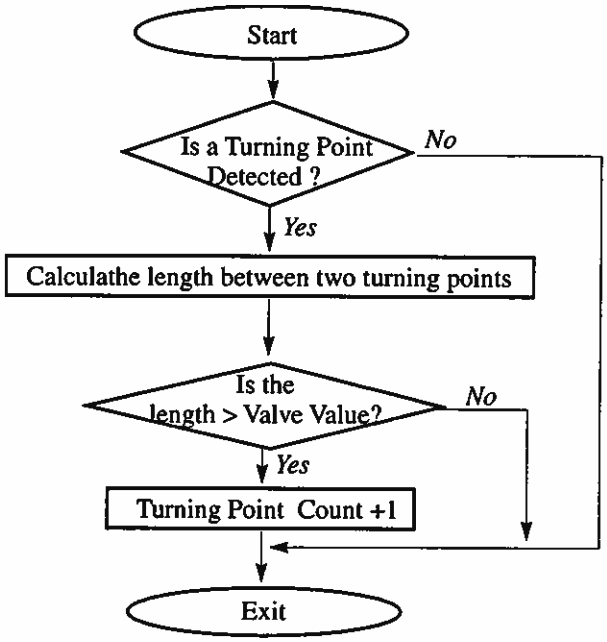


Fig 5 The Algorithm of Disturb Elimination

4. IMPLEMENTATION OF A SIMULATION SYSTEM

To verify the recognition algorithm and evaluate the system, a simulating system has been implemented on NeXT station, programming in Objective C. A window on the screen is designed as the board, and the mouse is used as the pen. User draws the gesture on the drawing area of the window by using mouse. As the recognition results, appropriate ASCII characters are displayed on a scroll text box placed on the bottom of the window. There are three control knobs under the drawing area, designed for adjusting the pen width, pen color and background color respectively. Fig 6 shows the architecture of the simulating system.

Moreover, there is some automatic test function specially designed to facilitate the recognition rate testing. To perform a test, the users can first draw a gesture as a sample to be tested, then click on the item "test" from the menu. This will initiate a test on the sample gesture. Afterward, each time the user draw a gesture, the system will compare it to the sample gesture and calculate the times of the failure comparison. Once 100 gesture has been drawn, the test will finish and a recognition rate will be calculated by

subtracting the failure times from the total test times and displayed on the scroll box. The users then could be able to test all the gestures one by one by repeating the same procedures described as above.

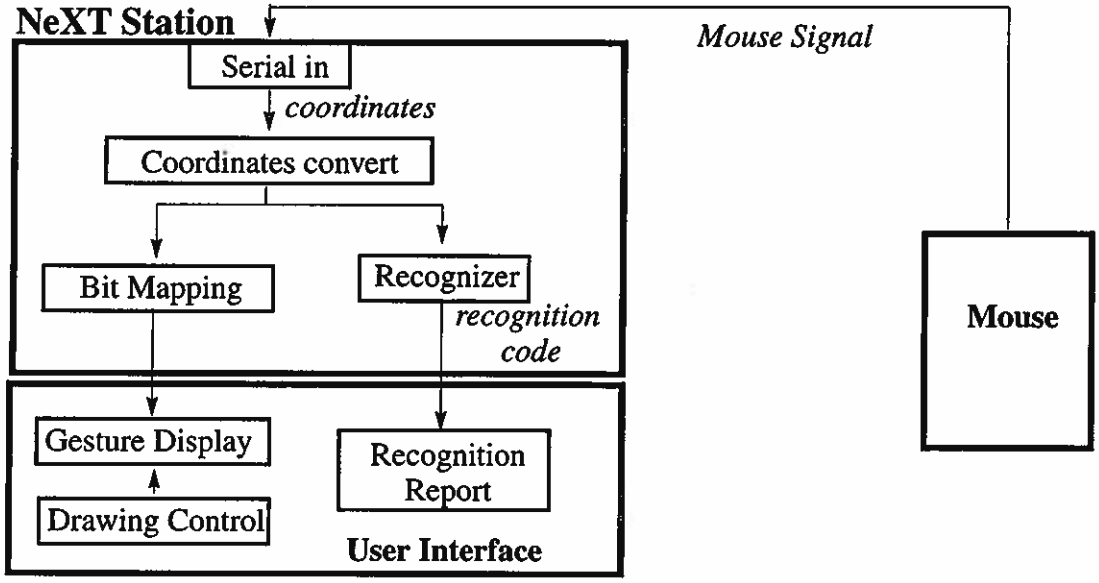


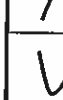
Fig 6. Architecture of a Simulating System

5. EVALUATION AND DISCUSSION

To investigate the gesture recognition rate, several tests have been made on the simulating system. The result of the tests is given in Tab 2, where there are 7 groups of tests altogether. Each group of test was taken under a valve value (VV) described in section 3.2. The VV has been set from 0 to 12 pixels incrementing in 2 pixels. As mentioned in chapter 4, each gesture would be tested for 100 times automatically, the total test times for each gesture should be: 7X100 = 700, and the grand total test times for all the gestures would be: 16X700 = 11,200. Attributed to the automatic test function, such a huge number of the tests could be done just in a few hours.

The results of the tests are satisfying enough. Some statistics based on the tests records have been taken and shown as Tab 3 and Tab 4. As it could be seen in Tab 3, most recognition rates are greater than 90% while VV is selected greater than 0. Among

the 7 groups, the highest recognition rate is up to 95.44 with VV = 8 pixels. The situation means that the recognition algorithm does really work, and the valve value VV is a sensitive factor for the recognition rate. The relationship between the recognition rate and the VV is shown in Fig 7.

Gesture	ASCII	Recognition Rate (%)						
		vv = 0	vv = 2	vv = 4	vv = 6	vv = 8	vv = 10	vv =12
	/	77	89	83	73	81	67	83
 (Tap)	·	81	89	95	95	83	88	81
	\	91	100	97	92	100	84	79
	(Enter)	74	99	100	95	100	100	97
	A	85	86	96	99	98	98	89
	V	76	96	97	94	99	89	93
	>	92	95	99	98	100	98	96
	<	61	85	91	100	97	95	89
	N	95	81	94	88	87	69	82
	n	78	93	99	98	100	99	97
	Z	63	88	85	93	100	92	77
	S	56	85	100	100	99	95	99
	M	60	86	75	94	99	98	85
	W	61	93	80	93	87	91	91
	}	68	95	93	94	99	95	85
	{	57	72	81	95	98	87	99

Tab 2. Test Report of Gesture Recognition Rate

Test Condition / recognition Rate(%)	vv = 0	vv = 2	vv = 4	vv = 6	vv = 8	vv = 10	vv =12
Maximun	95	100	100	100	100	100	99
Minimum	56	72	75	73	81	67	77
Average	73.44	89.50	91.56	93.81	95.44	90.31	88.88

Tab 3. The Recognition Rate Statistic (1)

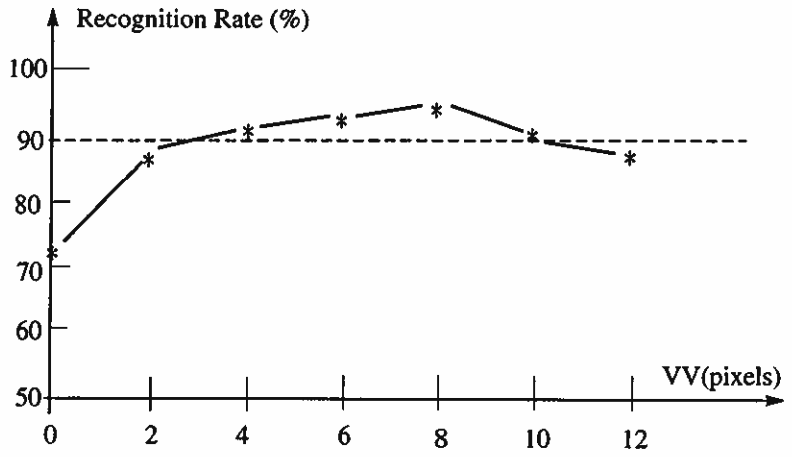


Fig 7 The relationship between Recognition Rate and Valve Value

A reasonable explanation could be given to the curved line in Fig 7. While the VV is smaller, some disturbs just greater than them could not be eliminated therefore leads to lower recognition rate. In the other hand, while the VV is too greater, some useful parts of the stroke will be over-eliminated therefor e causes misrecognition also. Hence, there should exists a best VV under which the gestures have a highest average of recognition rate, and the best VV maybe differ from one system to anothet. To be able to get as high as possible recognition rate for a real system, it is suggested that make the VV ajustable by creating some software knob which will allow the users to select the VV freely, similrly to the value and brightness control implemented in many computers.

The Tab 4 is another statistics from different angle. It lists the maximum, minimum and average of the recognition rate for each gesture. The averages are calculated from the data of 6 groups with VV greater than "0". The group with VV = 0 is ignored because that we will definitely need some disturb elimination processing so the data for VV = 0 is no longer our concern.

Recognition Code	Gesture	ASCII	Recognition Rate (%)		
			Maximun	Minimum	Average
0000		/	89	67	79.33
0001	 (Tap)	.	95	81	88.50
0010		\	100	84	92.00
0011		(Enter)	100	74	98.50
0100		A	99	86	94.33
0101		V	99	89	94.67
0110		>	100	95	97.67
0111		<	100	85	92.83
1000		N	94	69	83.50
1001		n	100	93	97.67
1010		Z	100	77	89.17
1011		S	100	85	96.33
1100		M	99	75	89.50
1101		W	93	80	89.16
1110		}	99	85	93.50
1111		{	99	72	88.67

Tab 4. The Recognition Rate Statistic (2)

Based on the data in Tab 4 which simply refer to the recognizabilities of the gestures, some curved lines have be drawn as Fig 8 so that it could be seen more clearly how the recognition rates are distributed among the gestures. Generally speaking, the gestures with only one turning point are relatively easier to be recognized, while the ones with too less (equals to "0") and too many turning points will be easier to be misrecognized. This situation makes common sense also. However, there are some exceptions as well. For example, the gesture "(Enter)" and "{" have higher recognition rates. The reason probably is related to the user's writing habits or the mechanical characteristics of the mouse. Exact explanations relies on the further exploration. But anyway, sine the avarage recognition rates of almost all the gestures could be promoted 90% above, except for the "/" which has a recognition rate just a little below 90%, it has no problem to have the conclusions that will be discussed below.

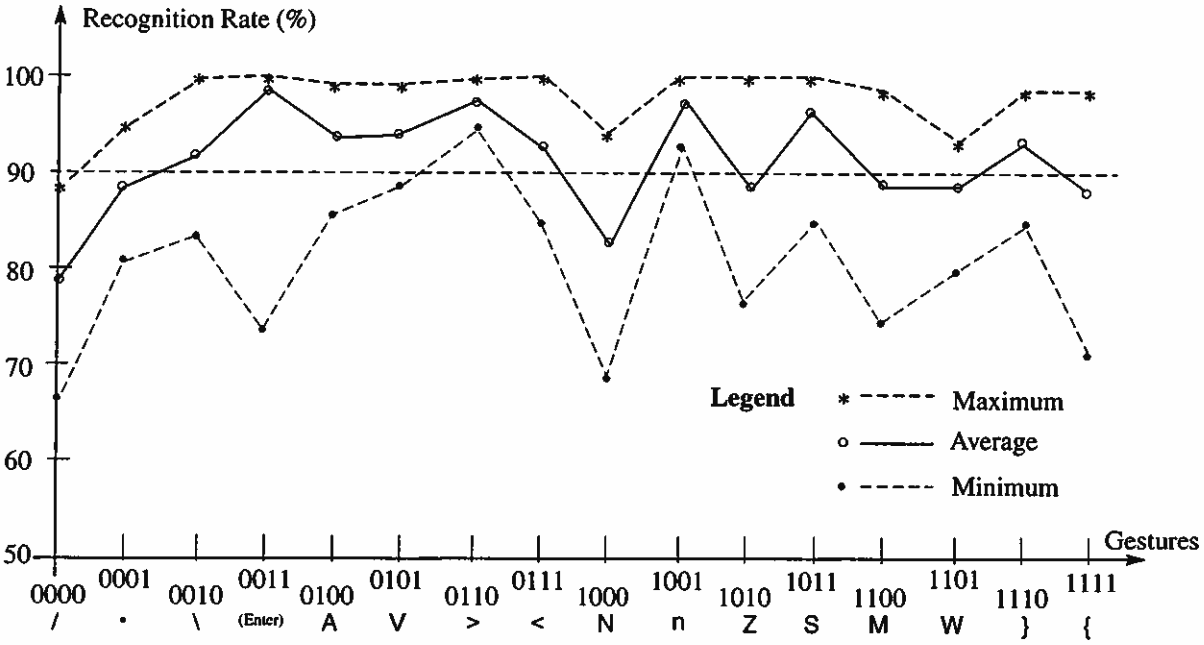


Fig 8. The Distribution of the Recognition Rates among the Gestures

6. CONCLUSIONS

The research on the single-stroke orientation-orient gesture system proves that the higher recognition rate is reachable by simple recognition algorithm under suitable

system design. It brings up a new kind of recognition algorithm -- orientation-orient recognition. Compared to the traditional ones, it is : orientation-orient rather than shape-orient; speed or time independent rather than speed or time dependent.

The recognition rate could be improved further by adding a new field -- let's call it *phase code*, to the recognition code. Reviewing the table of Tab 1., it could be found that there is another rule for the 16 gestures: the polarity of Δx in the first two rows remains unchanged (positive), while the polarity of Δy in the last two rows remains unchanged (negative). Hence, if the *phase code* for the first two rows is defined as 0, then the one for the last two rows could be defined as 1. The recognition algorithm for the new recognition code is just a little more complicated than the old one.

To make the gesture system more convincible, some now functional block for auto-calculating recognition rate is going to be added to the simulating system. It will allow drawing and recognizing the gestures by reading in data from particular files so that the users who have different writing habits could be able to to test the syste. Those data could be collected by using some real pan-based system.

The implementation of the orientation-orient gesture system in a real pen based computer system will be convenient. Its great simplicity helps to reduce the resource requirement, decrease its cost, and improve its responding speed. Considering that the drawing by pen should be easier to manipulate than mouse, the recognition rate in a real pen-based computer system should be even higher than the simulating one.

7. ACKNOWLEDGEMENTS

I would like to thank Dr. Takayuki Dan Kimura for his inviting me and supervising my research at his department, and for his help in writing this thesis.

I would like to thank Julie Chan for her assistance and technical support in my research work.

I also would like thank all the other members of Kumon Project Group, namely, Kastsuomo Uota, Makoto Umeda, Mahesh Tharamal, and Arun Eledath for thier goodness and cooperation in my daily work and life.

8. BIBLIOGRAPHY

[1] Kinmura, T.D., "Introduction to Dataflow-Based Visual Programming", Proceedings of 1994 IEEE Symposium on Visual Languages

[2] Kinmura, T.D., "Hyperflow: A Visual Programming Language for Pen Computers", Proceedings of 1992 IEEE Workshop on Visual Languages.

[3] Kimura, T.D., Choi, J.W., "A Visual language for Keyboardless Programming", Technical Report WUCS-86-6, Department of Computer Science, Washington University, St. Louis, MO, June 1986.

[4] Rhyne, James R. and Wolf, Catherine G., "Gestural Interface for Information Processing Application", Research Repot 12179 (#54544) 9/2/86, T.J.Watson Research Center, IBM corporation, P.O. Box 218, Yorktown Heights, NY 10598, September 29th, 1986

[5] Taysi, Burak M., "Gesture System for a Graph Editor", Thesis for the Degree of MS, Department of Computer Science, Washington University, St. Louis, MO, August 1992.